

GWD-I (Informational)

R. Aydt, University of Illinois at Urbana-Champaign  
D. Gunter, Lawrence Berkeley National Laboratory  
W. Smith, NASA Ames  
B. Tierney, Lawrence Berkeley National Laboratory  
V. Taylor, Northwestern University

GGF Grid Monitoring Architecture Working Group

February 2001  
Revised 03-December-2001

## **A Simple Case Study of a Grid Performance System**

Status of this Memo

This memo provides information to the Grid community regarding a simple performance monitoring scenario and an abstract implementation of a Grid performance system based on the Grid Monitoring Architecture (GMA) being developed by the Global Grid Forum Grid Monitoring Architecture Working Group. Distribution is unlimited.

Copyright Notice

Copyright © Global Grid Forum (2001). All Rights Reserved.

### **1. Abstract**

The Global Grid Forum Grid Monitoring Architecture Working Group has been developing a Grid Monitoring Architecture (GMA) that outlines one possible approach toward monitoring resources and applications in a Grid environment. This document presents a simple case study of a Grid monitoring system based on the GMA. It describes how the various system components would interact for a very basic monitoring scenario, and introduces the terminology and concepts presented in greater detail in other Working Group documents. It is hoped that this document will provide a basis for further discussion and prototype implementations of Grid monitoring systems.

Table of Contents

|     |   |    |
|-----|---|----|
| 1.  | Abstract.....                                     | 1  |
| 2.  | Introduction .....                                | 3  |
| 3.  | Scenario.....                                     | 3  |
| 4.  | Terminology .....                                 | 3  |
| 4.1 | Event, Event Type, and Event Data.....            | 3  |
| 4.2 | Event Schema .....                                | 4  |
| 4.3 | Producer.....                                     | 4  |
| 4.4 | Consumer .....                                    | 4  |
| 4.5 | Directory Service.....                            | 5  |
| 5.  | Implementation.....                               | 6  |
| 5.1 | Event Schema and Event Type Directory .....       | 6  |
| 5.2 | Event Producer Directory .....                    | 6  |
| 5.3 | Event Consumer Directory .....                    | 8  |
| 5.4 | Consumer/Producer Communication Established ..... | 8  |
| 5.5 | Producer Sends Event Data to Consumers .....      | 9  |
| 6.  | Summary.....                                      | 9  |
| 7.  | Open Issues .....                                 | 10 |
| 7.1 | Event Schema and Event Type Directory .....       | 10 |
| 7.2 | Event Producer Directory .....                    | 10 |
| 7.3 | Event Consumer Directory .....                    | 10 |
| 7.4 | Other Directory Services .....                    | 10 |
| 7.5 | Producer/Consumer Exchanges.....                  | 10 |
| 8.  | Security Considerations .....                     | 10 |
| 9.  | Glossary.....                                     | 10 |
| 10. | Author Information.....                           | 10 |
| 11. | Acknowledgements .....                            | 10 |
| 12. | Intellectual Property Statement.....              | 11 |
| 13. | Full Copyright Notice .....                       | 11 |
| 14. | References.....                                   | 11 |

## 2. Introduction

This document presents a simple case study of a Grid performance system based on the Grid Monitoring Architecture (GMA) being developed by the Global Grid Forum **Grid Monitoring Architecture** Working Group. It describes how the various system components would interact for a very basic monitoring scenario, and is intended to introduce people to the terminology and concepts presented in greater detail in other Working Group documents.

We believe that by focus ing on the simple case first, working group members can familiarize themselves with terminology and concepts, and productively join in the ongoing discussions of the group. In addition, prototype implementations of this basic scenario can be built to explore the feasibility of the proposed architecture and to expose possible shortcomings. Once the simple case is understood and agreed upon, complexities can be added incrementally as warranted by cases not addressed in the most basic implementation described here.

Some open issues and complex requirements that came up during the discussions of the simple scenario are briefly noted, but no attempt is made to address them in this document. We also do not address scalability, performance, or fault tolerance in this document. In the simple case presented here, it is assumed that all components have the necessary authorization to interact in the manner described. Clearly, authentication and authorization are two very important considerations in a real implementation of any Grid performance system.

## 3. Scenario

Ten workstations (*ws1* – *ws10*) are used as desktop systems by local users and are also available as compute engines for Grid applications. A monitor is running on each of these workstations to measure the CPU load every 30 seconds. The CPU load measurements are all forwarded to a central server (*srvr*) machine on the same local area network as the workstations. A process on the server makes the load information available to systems not located on the local network.

A system administrator for the *ws* workstations monitors the loads from her machine, *admins*, to ensure that there are no problems. Her machine, *admins*, is on a different network than the *ws* workstations.

Further, all of the load measurements are being archived by an archiving service on the machine *archiv*. The archival data is used by another program not discussed in the simple case study to analyze daily system load patterns and to identify time periods when the workstations are heavily utilized so that backups will not be scheduled during those times.

## 4. Terminology

In this section we define some of the basic terms used by the Global Grid Forum Working Group and relate them to the simple case study presented in this document.

### 4.1 Event, Event Type, and Event Data

An *event*<sup>\*</sup> is a structure containing one or more items of data that relate to one or more resources. Every event has an associated *event type* that uniquely identifies the structure for that particular event. The term *event data* refers to one or more of the items of data making up an event.

In the scenario described above, the machines *admins* and *archiv* are interested in events of type CPU\_LOAD. They want to receive CPU\_LOAD event data describing the load for systems *ws1* through *ws10*. Depending on the implementation, a single event may or may not contain information for all of the systems. In the implementation outlined below, an event contains the CPU load information for a single system.

## 4.2 Event Schema

An *event schema* describes the structure for a particular event.

In the basic scenario described in this document, a schema will be defined for the CPU\_LOAD event type.

Note that the data model for events has not yet been fully defined. In this document we restrict ourselves to a minimal data model of a named structure with one or more elements consisting of a tuple of (name, data type, description). << Dan, you said Value as the 3<sup>rd</sup> tuple element but in the schema it's description so that's what I used here. Am I mixing metaphors (or something worse) with this change? >>

## 4.3 Producer

A *producer* is a component that makes event data available to other components that are part of the Grid Monitoring Architecture. The producer speaks a standard protocol and generates event data in a standard format. It is possible that there will be multiple standard protocols and formats defined within the GMA. The producer may not be the originating source of the data – that source may or may not speak the same protocol and use the same event data format. The GMA is not concerned with defining the protocol(s) and format(s) used by the original sources of the performance data.

In our scenario, a process on *svr* is a producer and makes event data available to other components in the Grid performance system being described. Monitoring processes on *ws1* through *ws10* are the originating sources of the measurement data, but they are not producers.

## 4.4 Consumer

A *consumer* is a component in the Grid Monitoring Architecture that receives event data from a producer. It speaks a standard protocol and expects the event data to be in a standard format.

In the basic scenario described, processes on *admins* and *archiv* are consumers of the event data produced by *svr*. The *admins* consumer process will monitor the per-host CPU\_LOAD measurements. The *archiv* consumer process will write the event data to disk for later examination.

Note that although the *archives* process described in this scenario is a consumer process, that process may also function as a producer when data is extracted from the archive. The consumer/producer terms apply to functionality and interfaces -- it is possible for a single process (component) to act as a consumer **and** a producer, that is to have both consumer and producer

---

\* *Events*, as defined and used in this document, are implicitly *performance events*. We make no attempt to define or discuss other types of events.

functionality. <<Brian – I took a stab at addressing your concerns w/o adding “interface” everywhere. Is this good enough or can you or others think of better words or places to clarify this?>>

#### 4.5 Directory Service

A *directory service* is a searchable component in the Grid Monitoring Architecture used to store and forward information that is of general interest to other components in the system. The directory service can be queried through a variety of search mechanisms and returns information matching the specified selection criteria. The directory service may in practice be implemented as a set of distributed, interconnected individual directory services under the control of different organizations.

In the GMA, several distinct types of information will be stored in the directory service and we refer to the directories for each information type by a unique name. The actual implementation may place all entries in a single directory service, but conceptually we believe it is easiest to think of them as independent directories. Here we define only those directories that are necessary to implement the basic scenario.

##### 4.5.1 Event Type Directory

The *Event Type Directory* contains event schema for the various events in the system. The Event Type Directory does *not* contain actual events. For each event type there will be one schema in the Event Type Directory -- within the system all events of the same type must have the same structure.

The Event Type Directory can be searched by event type. It can also be searched by event element name, for example, “return all the event types that contain an element named *measurement*”.

To support the basic scenario described, the CPU\_LOAD schema must be included in the Event Type Directory.

##### 4.5.2 Event Producer Directory

The *Event Producer Directory* contains information about producers and the event types they provide.

All producer information in the Event Producer Dictionary is structured according to an *Event Producer Schema*. In contrast to the Event Type Directory, which contains the event schema but not the actual events, the Event Producer Directory *does* contain the actual producer information records and not just the schema for those records.

Consumers use the Event Producer Directory to locate producers of events they are interested in receiving. There are many possible ways a consumer might want to search for producers in the Event Producer Directory including: by event type, by producer, by host where the measurement originated, or by any number of other keys. The choice of what search keys should be supported is an open question.

For the basic scenario outlined in this document, the Event Producer Directory will contain one or more entries indicating that CPU\_LOAD event data for *ws1* through *ws10* is available from a producer on *srvr*.

##### 4.5.3 Event Consumer Directory

The *Event Consumer Directory* contains information about consumers, the event types they accept, and the services they provide.

All consumer information in the Event Consumer Dictionary is structured according to an *Event Consumer Schema*. As with the Event Producer Directory, the Event Consumer Directory *does* contain the actual consumer information records and not just the schema for those records.

Producers use the Event Consumer Directory to locate consumers that provide services of interest, or to find information on supported control and transport protocols and data formats for known consumers. As with the Event Producer Directory, the search keys that should be supported for the Event Consumer Directory remains an open question.

To support the basic scenario described in this document, the archival process on *archivsys* will register with the Event Consumer Directory as a consumer that accepts all event types and provides an archival service.

## 5. Implementation

In this section we describe, at a fairly high level, the steps necessary to implement the basic scenario with the Grid Monitoring Architecture. Through this description we hope to give the reader a clear idea of how the GMA components cooperate, and to provide a framework from which prototype implementations can be developed to test various protocols and formats.

### 5.1 Event Schema and Event Type Directory

To implement the basic Grid performance system described, we must first define the event schema for the CPU\_LOAD event. This schema will be stored in the Event Type Directory where it can be located and used to interpret data values in CPU\_LOAD events. We use a representation-independent format to define the schema here:

| Event Type   |                   | Event Description                      |
|--------------|-------------------|--|
| CPU_LOAD     |                   | CPU load measurement for a single host |
| Element Name | Element Data Type | Element Description                    |
| measurement  | double            | measured CPU load                      |
| hostname     | string            | host where measurement was taken       |
| timestamp    | ASCII timestamp   | time measurement was taken             |

**Figure 1: CPU\_LOAD Schema**

As defined, a *CPU\_LOAD* event has three data elements that contain the CPU load measurement, the host the measurement relates to, and the time the measurement was made  
 << Note that I took producer URL out of the table but it doesn't show up as a change >>

### 5.2 Event Producer Directory

The next step in the implementation process is for the producer, *srvr*, to add entries to the Event Producer Directory, advertising that it will provide CPU\_LOAD event data for *ws1*, *ws2*, ... *ws10*.

We have not yet reached a consensus on the contents of the Event Producer Directory entries, that is, the Event Producer Schema has not yet been set. We believe further discussion and experimentation are required to correctly identify an appropriate Event Producer Schema, and the version presented here should not be interpreted as a standard.

For the purpose of this simple case study we list the type of information that might be included in the Event Producer Directory entries. Two Event Producer Directory entries are shown, those for the CPU load data from *ws1* and *ws2*. Similar entries will exist for *ws3* through *ws10*.

| Field Name                | Value      |
|---------------------------|------------|
| <b>Producer_URL</b>       | svr:portXX |
| <b>Event_Type</b>         | CPU_LOAD   |
| <b>Host</b>               | ws1        |
| <b>Service</b>            | basic      |
| <b>Parameters</b>         | NONE       |
| <b>Filters</b>            | NONE       |
| <b>Access</b>             | OPEN       |
| <b>Control_Protocol</b>   | SOAP_TCP   |
| <b>Control_Data</b>       | SOAP_HTTP  |
| <b>Transport_Protocol</b> | SimpleXML  |
| <b>Transport_Data</b>     | XML, SDDF  |
| <b>Producer_URL</b>       | svr:portXX |
| <b>Event_Type</b>         | CPU_LOAD   |
| <b>Host</b>               | ws2        |
| <b>Service</b>            | basic      |
| <b>Parameters</b>         | NONE       |
| <b>Filters</b>            | NONE       |
| <b>Access</b>             | OPEN       |
| <b>Control_Protocol</b>   | SOAP_TCP   |
| <b>Control_Data</b>       | SOAP_HTTP  |
| <b>Transport_Protocol</b> | SimpleXML  |
| <b>Transport_Data</b>     | XML, SDDF  |

**Figure 2: Event Producer Directory Entries**

In the simple case study presented in this document, the consumer on *admins* is interested in CPU\_LOAD data for any of the *ws* machines. To support this scenario, the Event Producer Directory will be searched for entries with an *Event\_Type* of "CPU\_LOAD" and a *Host* of "ws1" through "ws10". The *Producer\_URL* field specifies where to contact the producer to receive events of interest.

The remaining Event Producer Directory fields are not explicitly used in this simple case study, but are included to show possible extensions that are discussed in other working group documents. *Service* could be used to indicate the types of queries that are supported by the producer interface. *Parameters* could be used to indicate that the producer would allow the consumer to control some producer variables, such as frequency of event record transmission. The *Filters* field could be used to indicate that the producer has some built-in filtering capabilities, such as sliding window average computations. The *Access* field is intended to provide different levels of access to the event data that is being produced – for example, make data available only to consumers within the same organization or make data available to anyone.

The *Control\_Protocol* and *Transport\_Protocol* fields could be used to specify which of several standard protocols the **producer understands** and allows for different control and data transport protocols. Sample protocols include SOAP\_TCP, SimpleXML, or SNMP. The *Control\_Data* and *Transport\_Data* fields could be used to specify which of several standard data formats **<< Here** Brian suggested adding "and protocols" but I don't think the *\_Data* fields do that - I think the *\_Protocol* fields do. Also, I'm unsure which fields "go" which directions... hence the highlights on "producer understands" above and "is capable of generating" here. Maybe we can discuss this in the call>> the **producer is capable of generating**, and allows for different data formats for control and data transport messages. Example data formats include SOAP\_HTTP, XML, ULM, SDDF, SNMP. A consumer may be fluent in a limited set of the possible protocols and formats and consequently would only consider connecting to producers that "speak" those protocols and formats.

### 5.3 Event Consumer Directory

Another step in the implementation process is for the archiving consumer on *archs* to advertise its existence. As with the Event Producer Directory entries, the Event Consumer Schema describing the contents of the Event Consumer Directory entries has not yet been finalized.

For the purpose of this simple case study we show the type of information that might be included in the Event Consumer Directory entries. An entry for the archiving consumer on *archs* is shown.

| Field Name                | Value                   |
|---------------------------|-------------------------|
| <b>Consumer_URL</b>       | archs:portYYY           |
| <b>Event_Type</b>         | *                       |
| <b>Service</b>            | archive                 |
| <b>Parameters</b>         | NONE                    |
| <b>Access</b>             | Producer=*.mydomain.edu |
| <b>Control_Protocol</b>   | SOAP_TCP                |
| <b>Control_Data</b>       | SOAP_HTTP               |
| <b>Transport_Protocol</b> | SimpleXML               |
| <b>Transport_Data</b>     | XML                     |

Figure 3: Event Consumer Directory Entry

The *Consumer\_URL* field specifies where to contact the consumer process, the *Event\_Type* field indicates the types of events the consumer is willing to accept, and the *Service* field shows the service or services the consumer provides. The other fields correspond to like-named fields in the Event Producer Schema. Note that values containing \*'s indicate wildcards.

### 5.4 Consumer/Producer Communication Established

Now that the directory service contains the event type schema, event producer information, and event consumer information, the Grid performance system is ready to share measurement information taken on resources in one part of the Grid with processes running on other systems in the Grid

In particular, for our simple case study the monitoring tool running on *admins* posts a query to the Event Producer Directory requesting any CPU\_LOAD events for machines *ws1* through *ws10*. The query returns ten matches, all with the same *Producer\_URL* contact values. Using the *Control\_Protocol* retrieved from the Event Producer Directory, the monitoring tool on *admins*

connects to the producer process at *svr:portXX*, and subscribes to the CPU\_LOAD events for *ws1*, *ws2*, ... *ws10*.

After starting up, the producer process on *svr* queries the Event Consumer Directory to find a consumer that offers archival services for CPU\_LOAD events, and the protocols and data formats the consumer understands. << Let's talk about this phrase in the call. Brian suggests it's unlikely, but this example was one Steve Fisher thought was plausible as to why a consumer directory might be needed. I did add earlier something on Dan's suggestion that you might check to find protocol, etc for known consumer. I guess I don't find this troubling as-is and would like to understand why others do before removing it >> Assuming *svr* is in "mydomain.edu", the entry for the archival service on *archivsys* is returned. At this point, the producer process on *svr* contacts the archival consumer process on *archivsys* and initiates a subscription to the producer's CPU\_LOAD events for *ws1* through *ws10*.

## 5.5 Producer Sends Event Data to Consumers

Once the subscriptions are in place, the producer sends CPU\_LOAD event data to the consumers until the subscriptions are cancelled.

The event data is sent using a well-defined protocol, which could be either the same as the control protocol or a different protocol negotiated during the connection process. The event data is sent in the format advertised in the Event Producer Directory. If the producer or consumer advertised that they understand multiple data formats, then the particular format is specified in the connection process.

Sample event data encoded in XML is shown here, with white space added for readability:

```
<CPU_LOAD>
  <measurement> 30.09 </measurement>
  <hostname> ws1 </hostname>
  <timestamp> 2001-01-30T20:33:05.003Z.001r.5a </timestamp>
  <producer> http://svr.mydomain.edu/producerXX </producer>
</CPU_LOAD>

<CPU_LOAD>
  <measurement> 22.98 </measurement>
  <hostname> ws9 </hostname>
  <timestamp> 2001-01-30T20:34:15.07Z.01r.5a </timestamp>
  <producer> http://svr.mydomain.edu/producerXX </producer>
</CPU_LOAD>
```

The monitoring tool receives the event data and updates the display for each host with the appropriate measurements. The archiving service receives the event data and writes it to the archive for later analysis by the backup-scheduling program.

## 6. Summary

We have described a very basic performance monitoring scenario in a Grid environment, defined terms used within the Global Grid Forum **Grid Monitoring Architecture** Working Group and related those to the scenario, and outlined at a fairly high level how the scenario could be implemented with the components defined in the Grid Monitoring Architecture. This basic scenario ignores many important and complex issues that are critical to a fully functional Grid performance system in the interest of presenting basic concepts and providing a starting point for discussion and prototype implementation experiments.

## 7. Open Issues

Many aspects of the GMA are not yet fully defined, including the event data model << is this the same as the event schema – would like to use consistent terms >>, directory service entries and search procedures, security mechanisms, and standard control and transport protocols and encodings.

Readers are encouraged to visit the GGF Grid Monitoring Architecture Working Group website at <http://www.didc.lbl.gov/GGF-PERF/GMA-WG> to view the latest GMA document and related proposals and prototypes. Interested parties are also welcome to participate in ongoing discussions regarding the GMA by attending GGF meeting and contributing to the Performance Area mailing list.

<< I'd like to take the details below and make them a working note or something within the WG so that we can go back later and see how/if we've addressed them all – or feel they are no longer important. – Ruth >>

## 8. Security Considerations

The document acknowledges that authorization and authentication are critical elements of a Grid monitoring system, but makes no attempt to address how the described system components would implement these security features.

## 9. Glossary

GMA                      Grid Monitoring Architecture, as defined by the Global Grid Forum Grid Monitoring Architecture Working Group.

## 10. Author Information

|  |   |   |
|--|---|---|
| Ruth A. Aydt<br>University of Illinois at Urbana-Champaign<br><a href="mailto:aydt@uiuc.edu">aydt@uiuc.edu</a><br>ph +1-217-333-8924<br>fx +1-217-244-6869 | Dan Gunter<br>Lawrence Berkeley National Laboratory<br><a href="mailto:dkgunter@lbl.gov">dkgunter@lbl.gov</a> | Warren Smith<br>NASA Ames<br><a href="mailto:wwsmith@arc.nasa.gov">wwsmith@arc.nasa.gov</a> |
| Brian L. Tierney<br>Lawrence Berkeley National Laboratory<br><a href="mailto:bltierney@lbl.gov">bltierney@lbl.gov</a>                                      | Valerie Taylor<br>Northwestern University<br><a href="mailto:taylor@ece.nwu.edu">taylor@ece.nwu.edu</a>       |   |

## 11. Acknowledgements

Thanks to Darcy Quesnel who participated in the initial SC2000 hallway discussions of the simple case study that led to this document. Thanks also to all members of the GGF Grid Monitoring Architecture Working Group, originally the Performance Working Group, who contributed to the initial and ongoing GMA discussions.

Ruth Aydt is supported in part by the Department of Energy under contract DOE W-7405-ENG-36 and by the National Science Foundation under Grant No. 9975020. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. Not sure if this disclaimer better here or after other funding..

Dan Gunter and Brian Tierney are supported by the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

## **12. Intellectual Property Statement**

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

## **13. Full Copyright Notice**

Copyright (C) Global Grid Forum (2001) All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."